Making Memory Swapping Practicable: Synergistic Coupling of SSD and Hard Disk for QoS-Aware Virtual Memory

Kei Davis, CCS-7;
Song Jiang, Wayne State University;
Xuechen Zhang, Georgia Institute of Technology;
Ke Liu, Wayne State University

Hard disk space has long been used to provide a virtual extension of main memory in order to allow computer programs with memory requirements that are greater than the available main memory to run. For large parallel machines, however, this mechanism is often not made available because of the excessive performance penalty—disk access is orders of magnitude slower than main memory. The relatively recent advent of the flash-memory-based solid-state drive (SSD), with access time several times faster than disk, would seem to be a technological improvement for this purpose. However, because SSD has sharply limited write endurance, a virtual memory extension based on SSD could quickly become unreliable. We have developed a quality of service (QoS)-aware system [1] that uses SSD and disk in tandem, exploiting the relative strengths of each and which provides a virtual memory extension that can be as fast or faster than SSD alone while minimizing the number of writes to SSD.

When a computer program's memory requirement, which typically grows during program execution, exceeds available memory, there are two possible outcomes depending on how the system is configured. The operating system on typical desktop computers or small servers transfers the least-recently-used pages of the memory image to hard disk to free up memory space, and transfers them back into memory (at the expense of moving other pages out) if they are later needed. This process is known as swapping, and the effective increase of available memory is known as a virtual memory extension. This is a cost-effective solution in the sense that the cost and energy consumption of disk, per unit of capacity, is orders of magnitude less than main memory (DRAM).

The price for this nearly free extra memory is performance—disk access, both in terms of latency (waiting time until transfer starts) and bandwidth (rate of data transfer once underway) is orders of magnitude slower than DRAM. In some scenarios this is acceptable–better that the program run slowly than quit running altogether. For large-scale scientific computing, however, this may not be acceptable—the computing platforms are expensive to purchase and to operate and maximizing their throughput is an economic imperative. As such, a swapping mechanism in the operating system is typically not available.

The relatively recent advent of the flash-memory-based solid-state drive (SSD), with much greater performance than disk for random and small accesses, suggests that by using SSD rather than disk the use of virtual memory extension would be acceptable and useful in more scenarios than it is currently. There are two problems with this simplistic approach. The first is that while SSD is much less expensive than DRAM in terms of capacity, it is more expensive than disk. The second is that SSD has a strictly limited life expectancy in terms of the number of write operations it can endure and, to make matters worse, the technological trend is towards greater capacity and lower cost at the expense of write endurance.

Our idea was to consider the relative strengths and weaknesses of disk and SSD, and to develop an algorithm that would distribute the load over an SSD-disk pair that would exploit their relative strengths to provide a virtual memory extension that is highly performant, cost effective, and with life expectancy comparable to disk.

Table 1. *Relative characteristics of DRAM, SSD, and disk.*

	DRAM	SSD	DISK
Power/GB	high	low	med
Cost/GB	high	med	low
Random/short access performance	high	med	low
Sequential access performance	high	med	med
Write endurance	unlimited	limited	unlimited

Table 1 compares the relevant characteristics of DRAM, SSD, and disk. With respect to power with other considerations aside, disk is a more efficient storage medium than DRAM, and SSD even more so. Regarding cost per unit of capacity, disk is less expensive than SSD, suggesting that the greater part of the memory extension would best reside on disk. SSD is much faster than disk for random (non-sequential) and

For more information contact Kei Davis at kei@lanl.gov.

Funding Acknowledgments
DOE, NNSA, Advanced Simulation and Computing Program

short accesses–this is because a non-sequential access will typically require a disk head movement and then wait for the data location on the spinning disk to appear under the disk head, requiring several milliseconds, before data starts to be read. The corresponding latency for SSD is tens of microseconds.

Where disk performance is more comparable to SSD is for long sequential accesses. Here the disk head can read or write data continuously as the disk spins underneath it, requiring only a short movement to the next concentric track after spanning the current one. In this scenario using disk instead of SSD is preferable for two reasons—cost/capacity (long reads or writes) and longevity (for writes).

The first step was to investigate whether representative memory-intensive applications generate memory-page access patterns that would warrant the use of disk to significantly reduce SSD writes without significant performance loss compared to SSD-only swapping. An important observation is that a sequence of page accesses need not be strictly sequential to be cast as a sequential access to disk. For example, if a sequence of accesses covers (possibly with small gaps) a contiguous span of memory pages it may directly cast as sequential if the pattern is detected the first time and is repeated. Extending this further, if any sequence of memory accesses is repeated it may be translated to a sequential access to disk. Instrumentation of representative applications shows that such patterns are common.

Based on these ideas and observations we developed a system in the Linux kernel, HybridSwap, and conducted an extensive performance evaluation using representative benchmarks. The behavior of HybridSwap is dynamically tunable–greater or lesser performance can be obtained by biasing traffic toward or away from the SSD, but with correspondingly greater or lesser write access to the SSD. There are two immediate consequences, the more obvious being that wear on the SSD can be controlled. The second is that we can provide quality-of-service functionality whereby bounds on the swapping penalty (within the capacity of the swapping system) may be specified for individual applications by effectively prioritizing their use of SSD.

Here we show a sample of results with HybridSwap at its default settings, chosen to yield application performance on par or slightly better than using SSD alone. Table 2 shows that HybridSwap reduces SSD writes by 16% to 40% for disparate benchmarks, and with run time reductions ranging from -0.8% to 7.1%.

Table 2. *Reduction in writes to SSD when running multiple competing instances of the Memcached (Memc), ImageMagick (Image), Matrix Inverse (Matrix), and Correlation Computation (CC) benchmarks.*

	Memc	Image	CC	Matrix
Write reduction	37%	40%	22%	16%
Performance improvement	-0.8%	4.6%	7.1%	0.5%

[1] Zhang, X. et al., "Synergistic Coupling of SSD and Hard Disk for QoS-aware Virtual Memory," *IEEE International Symposium on Performance Analysis Systems and Software (ISPASS)* (2013).